



(12)发明专利

(10)授权公告号 CN 103984586 B

(45)授权公告日 2017.02.08

(21)申请号 201410255584.7

G06F 3/06(2006.01)

(22)申请日 2014.06.10

(56)对比文件

(65)同一申请的已公布的文献号

KR 2005055341 A, 2005.06.13,

申请公布号 CN 103984586 A

CN 103293420 A, 2013.09.11,

CN 203351026 U, 2013.12.18,

(43)申请公布日 2014.08.13

审查员 陶治

(73)专利权人 哈尔滨工业大学

地址 150001 黑龙江省哈尔滨市南岗区西  
大直街92号

(72)发明人 高玉龙 张蔚 刘佳鑫 马永奎  
李然 张中兆

(74)专利代理机构 哈尔滨市松花江专利商标事  
务所 23109

代理人 张利明

(51)Int.Cl.

G06F 9/445(2006.01)

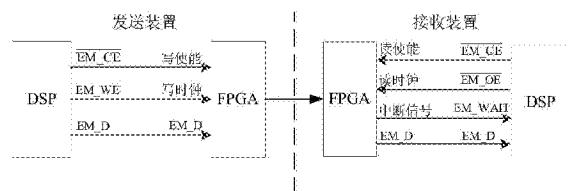
权利要求书2页 说明书6页 附图5页

(54)发明名称

嵌入式Linux系统下的EMIF与FPGA的接口驱  
动方法

(57)摘要

嵌入式Linux系统下的EMIF与FPGA的接口驱  
动方法，属于数据传输领域。解决了现有在Linux  
系统下的DSP的EMIF与FPGA接口无法实现图像数  
据传输的问题。该方法首先，实现EMIF驱动模块  
对驱动设备加载函数的加载和卸载函数的卸载，  
其次，通过EMIF驱动模块，实现发送装置中的DSP  
的EMIF端口对发送装置中的FPGA中FIFO进行写  
操作，再次，接收装置中的DSP接收到中断信号后，  
执行中断处理例程，唤醒读进程；最后，通过  
EMIF驱动模块，实现接收装置中的DSP的EMIF端  
口对接收装置中的FPGA中FIFO进行读操作。用于  
实现EMIF端口与FPGA的接口之间的数据通信。



1. 嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,它基于下述发送装置和接收装置实现的,发送装置的数据信号输出端与接收装置的数据信号输入端连接,

发送装置包括一个DSP和一个FPGA,其中,所述的DSP的写使能控制信号输出端与FPGA的写使能信号输入端连接,DSP的写时钟信号输出端与FPGA的写时钟信号输入端连接,DSP的数据信号输出端与FPGA的数据信号输入端连接,发送装置中FPGA的数据信号输出端作为发送装置的数据信号输出端;

接收装置包括一个DSP和一个FPGA,其中,所述的FPGA的读使能控制信号输入端与DSP的读使能信号输出端连接,FPGA的读时钟信号输入端与DSP的读时钟信号输出端连接,FPGA的中断触发信号输出端与DSP的中断信号输入端连接,FPGA的数据信号输出端与DSP的数据信号输入端连接,接收装置中FPGA的数据信号输入端作为接收装置的数据信号输入端;

发送装置中的DSP和接收装置中的DSP均为含有EMIF端口的DSP,

发送装置中的DSP和接收装置中的DSP均嵌入了Linux操作系统和EMIF驱动模块,在Linux操作系统下,EMIF驱动模块用于实现DSP与FPGA的通信,

嵌入在发送装置中的DSP内部的EMIF驱动模块实现对FPGA接口驱动的方法为,

步骤1:实现EMIF驱动模块对驱动设备加载函数的加载和该驱动设备卸载函数的卸载,

步骤2:通过EMIF驱动模块,将FPGA的FIFO映射为DSP的一个内存地址,将对FIFO的写操作映射为对所述的内存地址的写操作,实现发送装置中的DSP的EMIF端口对发送装置中的FPGA中FIFO进行写操作,

步骤3:接收装置中的DSP接收到中断触发信号后,执行中断处理例程,使接收装置中的DSP的EMIF端口从接收装置中的FPGA的FIFO中读出指定数目的字节,并存储到Linux操作系统内核的缓存区中,唤醒读进程;

步骤4:通过EMIF驱动模块,将FPGA的FIFO映射为DSP的一个内存地址,将对FIFO的读操作映射为对所述的内存地址的读操作,实现接收装置中的DSP的EMIF端口对接收装置中的FPGA中FIFO进行读操作,即完成EMIF端口与FPGA的接口之间的数据通信。

2. 根据权利要求1所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,所述的实现EMIF驱动模块对驱动设备加载函数的加载和该驱动设备卸载函数的卸载中,

EMIF驱动模块对驱动设备加载函数的加载过程为:

在EMIF驱动模块对驱动设备进行驱动的过程中,首先,对设备号进行申请,并将EMIF驱动模块注册到Linux操作系统中,Linux操作系统通过设备号确定该设备对应的EMIF驱动模块,完成EMIF驱动模块对驱动设备加载函数的加载,

EMIF驱动模块对驱动设备卸载函数的卸载过程为:

将设备号进行释放,并将驱动设备对应的EMIF驱动模块所占用的内存进行释放,并返回I/O资源给系统,完成EMIF驱动模块对驱动设备卸载函数的卸载。

3. 根据权利要求1或2所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,所述的实现发送装置中的DSP的EMIF端口对发送装置中的FPGA中FIFO进行写操作的具体过程为,

通过写操作write()函数将数据从发送装置中的DSP中复制到发送装置中的FPGA的FIFO中。

4.根据权利要求3所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,所述的实现接收装置中的DSP的EMIF端口对接收装置中的FPGA中FIFO进行读操作的具体过程为,

通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间。

5.根据权利要求1、2或4所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,所述的接收装置中的DSP接收到中断触发信号后,执行中断处理例程的具体过程为,

步骤4-1:清除中断标志,执行步骤4-2,

步骤4-2:采用I/O操作函数将接收装置的FPGA中FIFO的数据复制到接收装置的DSP的内核空间的缓存区,执行步骤4-3,

步骤4-3:判断内核空间的缓存区是否满一半,结果为是,执行步骤4-4,结果为否,执行步骤4-5,

步骤4-4:唤醒阻塞的用户空间的读进程,执行步骤4-5;

步骤4-5:中断返回。

6.根据权利要求4所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,所述的通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间的具体过程为:

步骤5-1:判断接收装置中DSP的内核缓存区是否为空,结果为是,执行步骤5-2,结果为否,执行步骤5-3,

步骤5-2:读进程休眠,直到缓存区不为空时,执行步骤5-3,

步骤5-3:唤醒读进程,执行步骤5-4,

步骤5-4:通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间,执行步骤5-5,

步骤5-5:释放信号量。

7.根据权利要求3所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,所述的通过写操作write()函数将数据从发送装置中的DSP中复制到发送装置中的FPGA的FIFO中的具体过程为:

步骤3-1:在发送装置中DSP的内核空间申请缓存区后,执行步骤3-2,

步骤3-2:将用户空间的数据复制到内核空间的缓存区,执行步骤3-3,

步骤3-3:通过写操作write()函数将内核空间缓存区内容复制到发送装置中的FPGA的FIFO中,执行步骤3-4;

步骤3-4:释放信号量。

8.根据权利要求1、2、4、6或7所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,其特征在于,所述的发送装置中的DSP和接收装置中的DSP均采用型号为TMS320DM365的芯片实现。

## 嵌入式Linux系统下的EMIF与FPGA的接口驱动方法

### 技术领域

[0001] 本发明属于数据传输领域。

### 背景技术

[0002] 随着生活水平的提高和科技水平的发展,人们对图像处理的要求越来越高,如视频会议、视频监控、视频通话等。但是,在达芬奇(Davinci)技术出现之前,针对不同的需要,开发者需要掌握不同的图像处理芯片和软件的开发知识,这大大增加了新产品开发的难度和开发周期。针对上述情况,TI公司在DSP技术的基础上,结合ARM、图像编解码标准, TI编程规范等提出了达芬奇技术架构。Davinci技术架构大大降低了图像处理的复杂度,缩短了产品的研发周期,并且还能提高产品性能。

[0003] 随着DSP处理速度提高和开发工具的完善,其应用范围越来越广泛。由于DSP的内部RAM有限,所以要实现大量数据的采集和处理,需要对DSP的RAM进行扩展。

[0004] 另外,在无线通信的过程中,传输的往往是串行数据,但是由于DSP本身硬件原理和指令的特点,使得DSP不能与无线通信设备进行无缝连接。

[0005] 通过DSP的EMIF接口与FPGA进行连接,可以使得DSP获得较大的外部存储空间。同时,由于FPGA具有编程灵活性的特点,使得DSP能够与众多的模块设备进行无缝连接,这样就大大拓展了DSP的应用范围。

### 发明内容

[0006] 本发明是为了解决现有在Linux系统下的DSP的EMIF与FPGA接口无法实现图像数据传输的问题,本发明提供了一种嵌入式Linux系统下的EMIF与FPGA的接口驱动方法。

[0007] 嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,它基于下述发送装置和接收装置实现的,发送装置的数据信号输出端与接收装置的数据信号输入端连接,

[0008] 发送装置包括一个DSP和一个FPGA,其中,所述的DSP的写使能控制信号输出端与FPGA的写使能信号输入端连接,DSP的写时钟信号输出端与FPGA的写时钟信号输入端连接,DSP的数据信号输出端与FPGA的数据信号输入端连接,发送装置中FPGA的数据信号输出端作为发送装置的数据信号输出端;

[0009] 接收装置包括一个DSP和一个FPGA,其中,所述的FPGA的读使能控制信号输入端与DSP的读使能信号输出端连接,FPGA的读时钟信号输入端与DSP的读时钟信号输出端连接,FPGA的中断触发信号输出端与DSP的中断信号输入端连接,FPGA的数据信号输出端与DSP的数据信号输入端连接,接收装置中FPGA的数据信号输入端作为接收装置的数据信号输入端;

[0010] 发送装置中的DSP和接收装置中的DSP均为含有EMIF端口的DSP,

[0011] 发送装置中的DSP和接收装置中的DSP均嵌入了Linux操作系统和EMIF驱动模块,在Linux操作系统下,EMIF驱动模块用于实现DSP与FPGA的通信,

[0012] 嵌入在发送装置中的DSP内部的EMIF驱动模块实现对FPGA接口驱动的方法为,

- [0013] 步骤1:实现EMIF驱动模块对驱动设备加载函数的加载和该驱动设备卸载函数的卸载，
- [0014] 步骤2:通过EMIF驱动模块,将FPGA的FIFO映射为DSP的一个内存地址,将对FIFO的写操作映射为对所述的内存地址的写操作,实现发送装置中的DSP的EMIF端口对发送装置中的FPGA中FIFO进行写操作，
- [0015] 步骤3:接收装置中的DSP接收到中断触发信号后,执行中断处理例程,使接收装置中的DSP的EMIF端口从接收装置中的FPGA的FIFO中读出指定数目的字节,并存储到Linux操作系统内核的缓存区中,唤醒读进程;
- [0016] 步骤4:通过EMIF驱动模块,将FPGA的FIFO映射为DSP的一个内存地址,将对FIFO的读操作映射为对所述的内存地址的读操作,实现接收装置中的DSP的EMIF端口对接收装置中的FPGA中FIFO进行读操作,即完成EMIF端口与FPGA的接口之间的数据通信。
- [0017] 所述的实现EMIF驱动模块对驱动设备加载函数的加载和该驱动设备卸载函数的卸载中,EMIF驱动模块对驱动设备加载函数的加载过程为:
- [0018] 在EMIF驱动模块对驱动设备进行驱动的过程中,首先,对设备号进行申请,并将EMIF驱动模块注册到Linux操作系统中,Linux操作系统通过设备号确定该设备对应的EMIF驱动模块,完成EMIF驱动模块对驱动设备加载函数的加载,
- [0019] EMIF驱动模块对驱动设备卸载函数的卸载过程为:
- [0020] 将设备号进行释放,并将驱动设备对应的EMIF驱动模块所占用的内存进行释放,并返回I/O资源给系统,完成EMIF驱动模块对驱动设备卸载函数的卸载。
- [0021] 所述的实现发送装置中的DSP的EMIF端口对发送装置中的FPGA中FIFO进行写操作的具体过程为,
- [0022] 通过写操作write()函数将数据从发送装置中的DSP中复制到发送装置中的FPGA的FIFO中。
- [0023] 所述的实现接收装置中的DSP的EMIF端口对接收装置中的FPGA中FIFO进行读操作的具体过程为,
- [0024] 通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间。
- [0025] 所述的接收装置中的DSP接收到中断触发信号后,执行中断处理例程的具体过程为,
- [0026] 步骤4-1:清除中断标志,执行步骤4-2,
- [0027] 步骤4-2:采用I/O操作函数将接收装置的FPGA中FIFO的数据复制到接收装置的DSP的内核空间的缓存区,执行步骤4-3,
- [0028] 步骤4-3:判断内核空间的缓存区是否满一半,结果为是,执行步骤4-4,结果为否,执行步骤4-5,
- [0029] 步骤4-4:唤醒阻塞的用户空间的读进程,执行步骤4-5;
- [0030] 步骤4-5:中断返回。
- [0031] 所述的通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间的具体过程为:
- [0032] 步骤5-1:判断接收装置中DSP的内核缓存区是否为空,结果为是,执行步骤5-2,结果为否,执行步骤5-3,

- [0033] 步骤5-2:读进程休眠,直到缓存区不为空时,执行步骤5-3,
- [0034] 步骤5-3:唤醒读进程,执行步骤5-4,
- [0035] 步骤5-4:通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间,执行步骤5-5,
- [0036] 步骤5-5:释放信号量。
- [0037] 所述的通过写操作write()函数将数据从发送装置中的DSP中复制到发送装置中的FPGA的FIFO中的具体过程为:
- [0038] 步骤3-1:在发送装置中DSP的内核空间申请缓存区后,执行步骤3-2,
- [0039] 步骤3-2:将用户空间的数据复制到内核空间的缓存区,执行步骤3-3,
- [0040] 步骤3-3:通过写操作write()函数将内核空间缓存区内容复制到发送装置中的FPGA的FIFO中,执行步骤3-4;
- [0041] 步骤3-4:释放信号量。
- [0042] 所述的发送装置中的DSP和接收装置中的DSP均采用型号为TMS320DM365的芯片实现。整个系统中,设置3个测试点A、B、C,进行测试,测试点A、B、C在系统中的位置如图4所示。
- [0043] 在测试过程中,首先发送装置中TMS320DM365向该装置中FPGA发送数据,然后发送装置中FPGA将数据发送给接收装置中FPGA,最后接收装置中FPGA再将数据发送给接收装置中TMS320DM365。其中A、B和C三个测试点的FPGA波形分别如图6、7和8所示,通过比较TMS320DM365发送与接收的数据,证明TMS320DM365与FPGA之间的接口设计是正确的。
- [0044] 本发明带来的有益效果是,实现了Linux系统下的DSP的EMIF与FPGA接口实现图像数据传输。

## 附图说明

- [0045] 图1为具体实施方式一所述的发送装置和接收装置的原理示意图;
- [0046] 图2为具体实施方式六中,通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间的流程图;
- [0047] 图3为具体实施方式七中,通过写操作write()函数将数据从发送装置中的DSP中复制到发送装置中的FPGA的FIFO中的流程图;
- [0048] 图4为具体实施方式八中,测试点A、B、C在发送装置和接收装置中的位置关系示意图;
- [0049] 图5为具体实施方式五中,所述的接收装置中的DSP接收到中断触发信号后,执行中断处理例程的流程图;
- [0050] 图6为测试点A的波形示意图;
- [0051] 图7为测试点B的波形示意图;
- [0052] 图8为测试点C的波形示意图。

## 具体实施方式

- [0053] 具体实施方式一:参见图1说明本实施方式,本实施方式所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法,它基于下述发送装置和接收装置实现的,发送装置的数据信号输出端与接收装置的数据信号输入端连接,

[0054] 发送装置包括一个DSP和一个FPGA，其中，所述的DSP的写使能控制信号输出端与FPGA的写使能信号输入端连接，DSP的写时钟信号输出端与FPGA的写时钟信号输入端连接，DSP的数据信号输出端与FPGA的数据信号输入端连接，发送装置中FPGA的数据信号输出端作为发送装置的数据信号输出端；

[0055] 接收装置包括一个DSP和一个FPGA，其中，所述的FPGA的读使能控制信号输入端与DSP的读使能信号输出端连接，FPGA的读时钟信号输入端与DSP的读时钟信号输出端连接，FPGA的中断触发信号输出端与DSP的中断信号输入端连接，FPGA的数据信号输出端与DSP的数据信号输入端连接，接收装置中FPGA的数据信号输入端作为接收装置的数据信号输入端；

[0056] 发送装置中的DSP和接收装置中的DSP均为含有EMIF端口的DSP，

[0057] 发送装置中的DSP和接收装置中的DSP均嵌入了Linux操作系统和EMIF驱动模块，在Linux操作系统下，EMIF驱动模块用于实现DSP与FPGA的通信，

[0058] 嵌入在发送装置中的DSP内部的EMIF驱动模块实现对FPGA接口驱动的方法为，

[0059] 步骤1：实现EMIF驱动模块对驱动设备加载函数的加载和该驱动设备卸载函数的卸载，

[0060] 步骤2：通过EMIF驱动模块，将FPGA的FIFO映射为DSP的一个内存地址，将对FIFO的写操作映射为对所述的内存地址的写操作，实现发送装置中的DSP的EMIF端口对发送装置中的FPGA中FIFO进行写操作，

[0061] 步骤3：接收装置中的DSP接收到中断触发信号后，执行中断处理例程，使接收装置中的DSP的EMIF端口从接收装置中的FPGA的FIFO中读出指定数目的字节，并存储到Linux操作系统内核的缓存区中，唤醒读进程；

[0062] 步骤4：通过EMIF驱动模块，将FPGA的FIFO映射为DSP的一个内存地址，将对FIFO的读操作映射为对所述的内存地址的读操作，实现接收装置中的DSP的EMIF端口对接收装置中的FPGA中FIFO进行读操作，即完成EMIF端口与FPGA的接口之间的数据通信。

[0063] 具体实施方式二：本实施方式与具体实施方式一所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法的区别在于，所述的实现EMIF驱动模块对驱动设备加载函数的加载和该驱动设备卸载函数的卸载中，

[0064] EMIF驱动模块对驱动设备加载函数的加载过程为：

[0065] 在EMIF驱动模块对驱动设备进行驱动的过程中，首先，对设备号进行申请，并将EMIF驱动模块注册到Linux操作系统中，Linux操作系统通过设备号确定该设备对应的EMIF驱动模块，完成EMIF驱动模块对驱动设备加载函数的加载，

[0066] EMIF驱动模块对驱动设备卸载函数的卸载过程为：

[0067] 将设备号进行释放，并将驱动设备对应的EMIF驱动模块所占用的内存进行释放，并返回I/O资源给系统，完成EMIF驱动模块对驱动设备卸载函数的卸载。具体实施方式三：本实施方式与具体实施方式一或二所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法的区别在于，所述的实现发送装置中的DSP的EMIF端口对发送装置中的FPGA中FIFO进行写操作的具体过程为，

[0068] 通过写操作write()函数将数据从发送装置中的DSP中复制到发送装置中的FPGA的FIFO中。

[0069] 具体实施方式四:本实施方式与具体实施方式三所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法的区别在于,所述的实现接收装置中的DSP的EMIF端口对接收装置中的FIFO进行读操作的具体过程为,

[0070] 通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间。

[0071] 具体实施方式五:参见图5说明本实施方式,本实施方式与具体实施方式一、二或四所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法的区别在于,所述的接收装置中的DSP接收到中断触发信号后,执行中断处理例程的具体过程为,

[0072] 步骤4-1:清除中断标志,执行步骤4-2,

[0073] 步骤4-2:采用I/O操作函数将接收装置的FIFO的数据复制到接收装置的DSP的内核空间的缓存区,执行步骤4-3,

[0074] 步骤4-3:判断内核空间的缓存区是否满一半,结果为是,执行步骤4-4,结果为否,执行步骤4-5,

[0075] 步骤4-4:唤醒阻塞的用户空间的读进程,执行步骤4-5;

[0076] 步骤4-5:中断返回。

[0077] 具体实施方式六:参见图2说明本实施方式,本实施方式与具体实施方式四所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法的区别在于,所述的通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间的具体过程为:

[0078] 步骤5-1:判断接收装置中DSP的内核缓存区是否为空,结果为是,执行步骤5-2,结果为否,执行步骤5-3,

[0079] 步骤5-2:读进程休眠,直到缓存区不为空时,执行步骤5-3,

[0080] 步骤5-3:唤醒读进程,执行步骤5-4,

[0081] 步骤5-4:通过读操作read()函数将接收装置中DSP的内核缓存区的数据复制到用户空间,执行步骤5-5,

[0082] 步骤5-5:释放信号量。

[0083] 具体实施方式七:参见图3说明本实施方式,本实施方式与具体实施方式三所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法的区别在于,所述的通过写操作write()函数将数据从发送装置中的DSP中复制到发送装置中的FPGA的FIFO中的具体过程为:

[0084] 步骤3-1:在发送装置中DSP的内核空间申请缓存区后,执行步骤3-2,

[0085] 步骤3-2:将用户空间的数据复制到内核空间的缓存区,执行步骤3-3,

[0086] 步骤3-3:通过写操作write()函数将内核空间缓存区内容复制到发送装置中的FPGA的FIFO中,执行步骤3-4;

[0087] 步骤3-4:释放信号量。

[0088] 具体实施方式八:本实施方式与具体实施方式一、二、四、六或七所述的嵌入式Linux系统下的EMIF与FPGA的接口驱动方法的区别在于,所述的发送装置中的DSP和接收装置中的DSP均采用型号为TMS320DM365的芯片实现。

[0089] 本实施方式中,发送装置的TMS320DM365与FPGA的硬件连接中,EM\_CE是TMS320DM365输出的片选信号,作为FPGA中FIFO的写使能,且低电平有效;EM\_D是TMS320DM365的双向数据线,此模式下为数据输出总线,与FPGA输入FIFO相连;EM\_WE为

TMS320DM365输出的写使能信号,作为FPGA中FIFO的写时钟,由EMIF接口的写时序可知,数据会在EM\_WE的上升沿被存入发送装置的FPGA的FIFO中;

[0090] 接收装置的TMS320DM365与FPGA的硬件连接中,EM\_CE是TMS320DM365输出的片选信号,作为FPGA中FIFO的读使能,且低电平有效;EM\_D是TMS320DM365的双向数据线,此模式下为数据输入总线,与FPGA输出FIFO相连;EM\_OE为TMS320DM365输出的读使能信号,作为FPGA中FIFO的读时钟。由EMIF接口的读时序可知,在EM\_OE的上升沿将视频数据通过EMIF接口由FPGA的FIFO写入TMS320DM365中。

[0091] 本实施方式中,整个系统中,设置3个测试点A、B、C,进行测试,测试点A、B、C在系统中的位置如图4所示。

[0092] 在测试过程中,首先发送装置中TMS320DM365向该装置中FPGA发送数据,然后发送装置中FPGA将数据发送给接收装置中FPGA,最后接收装置中FPGA再将数据发送给接收装置中TMS320DM365。其中A、B和C三个测试点的FPGA波形分别如图6、7和8所示,通过比较TMS320DM365发送与接收的数据,证明TMS320DM365与FPGA之间的接口设计是正确的。

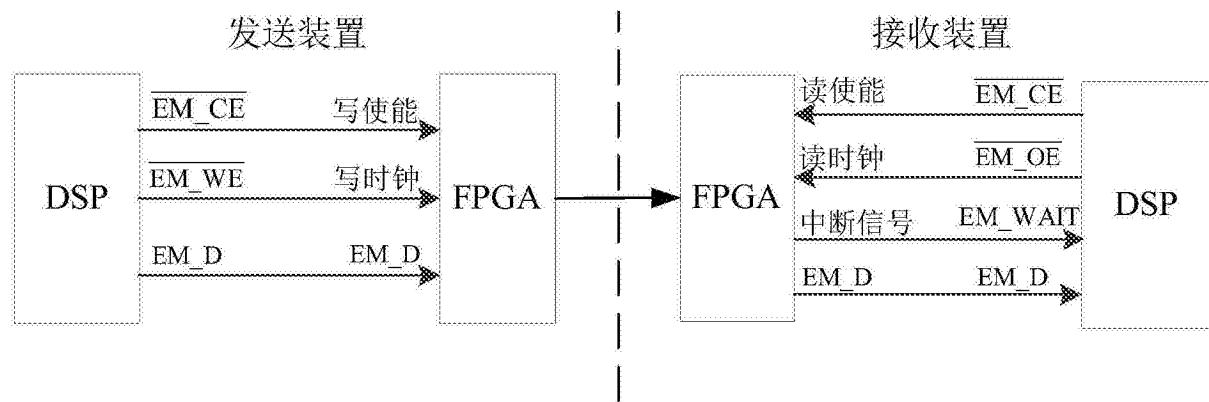


图1

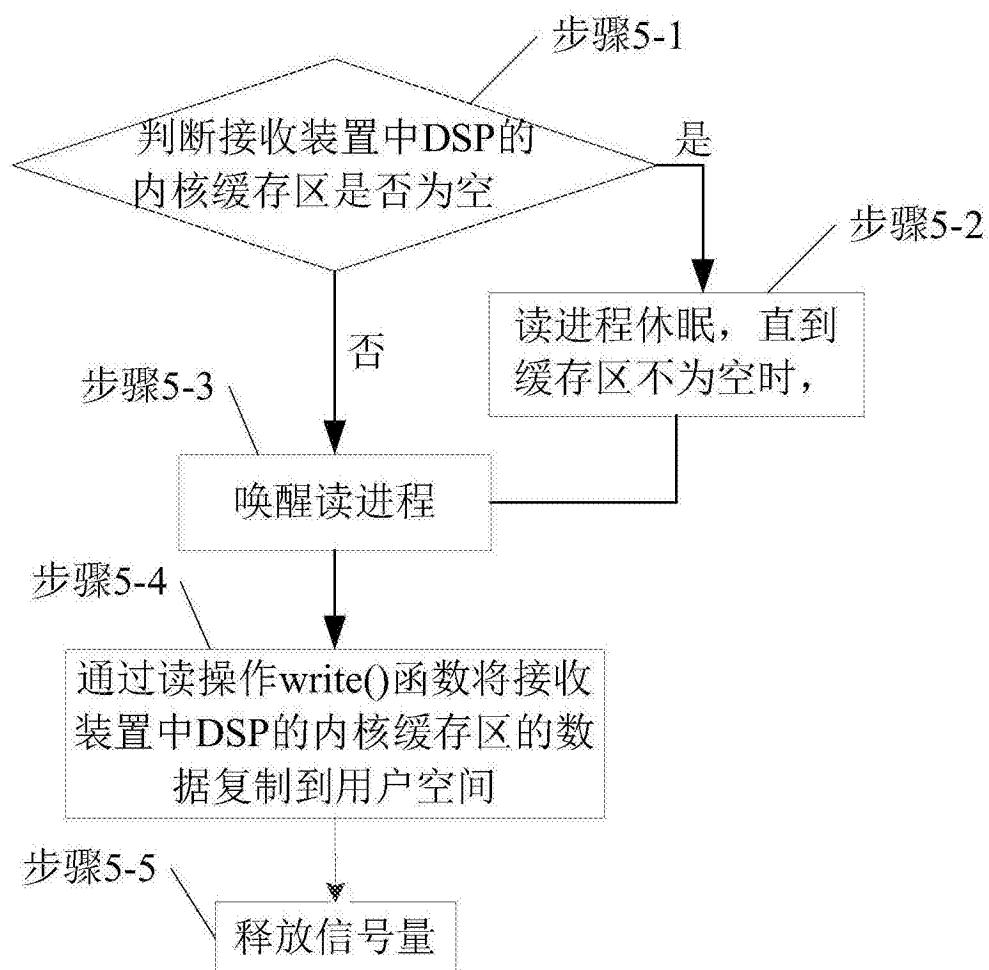


图2

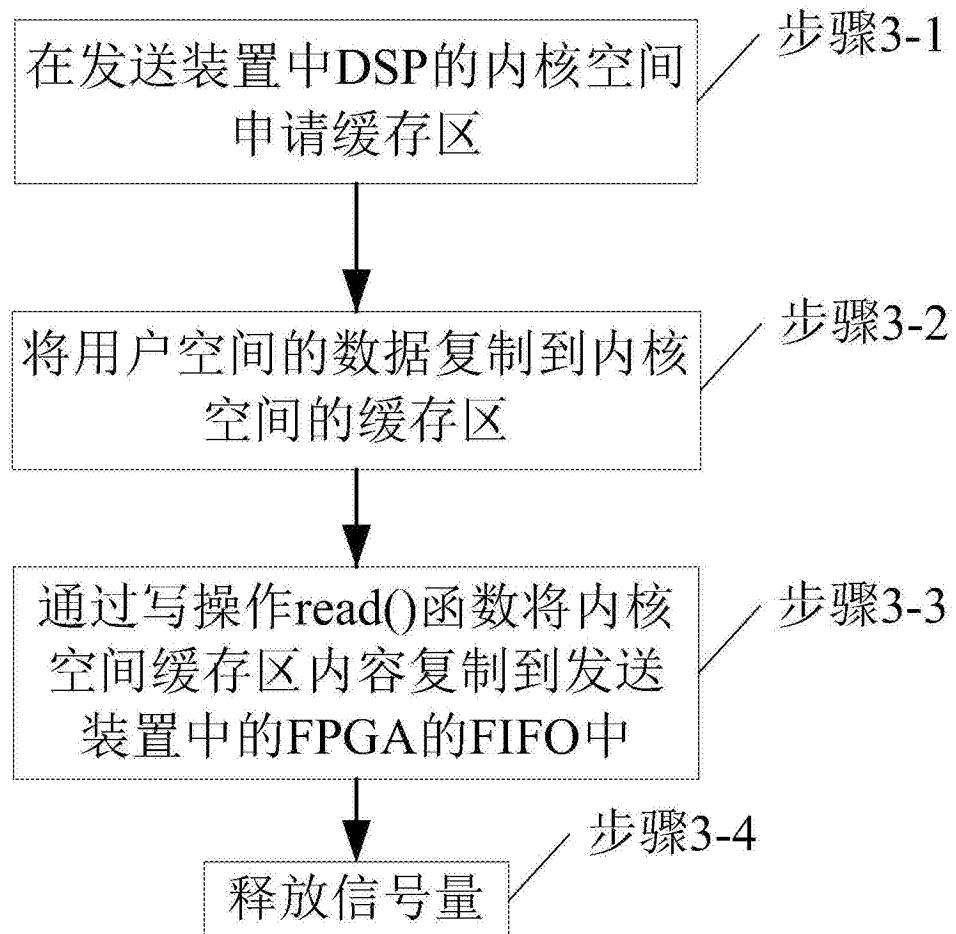


图3

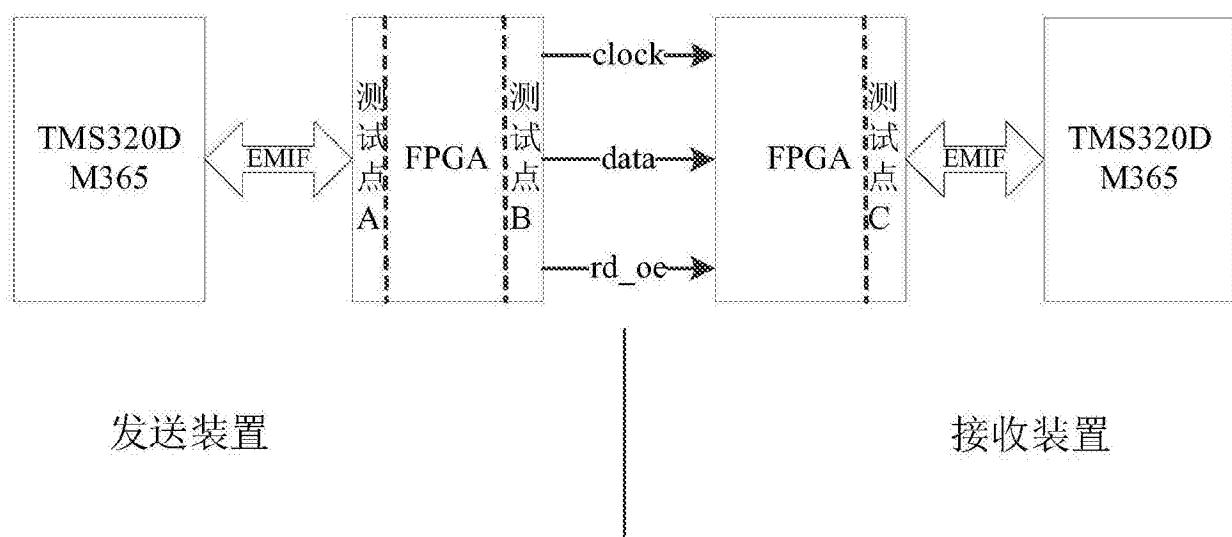


图4

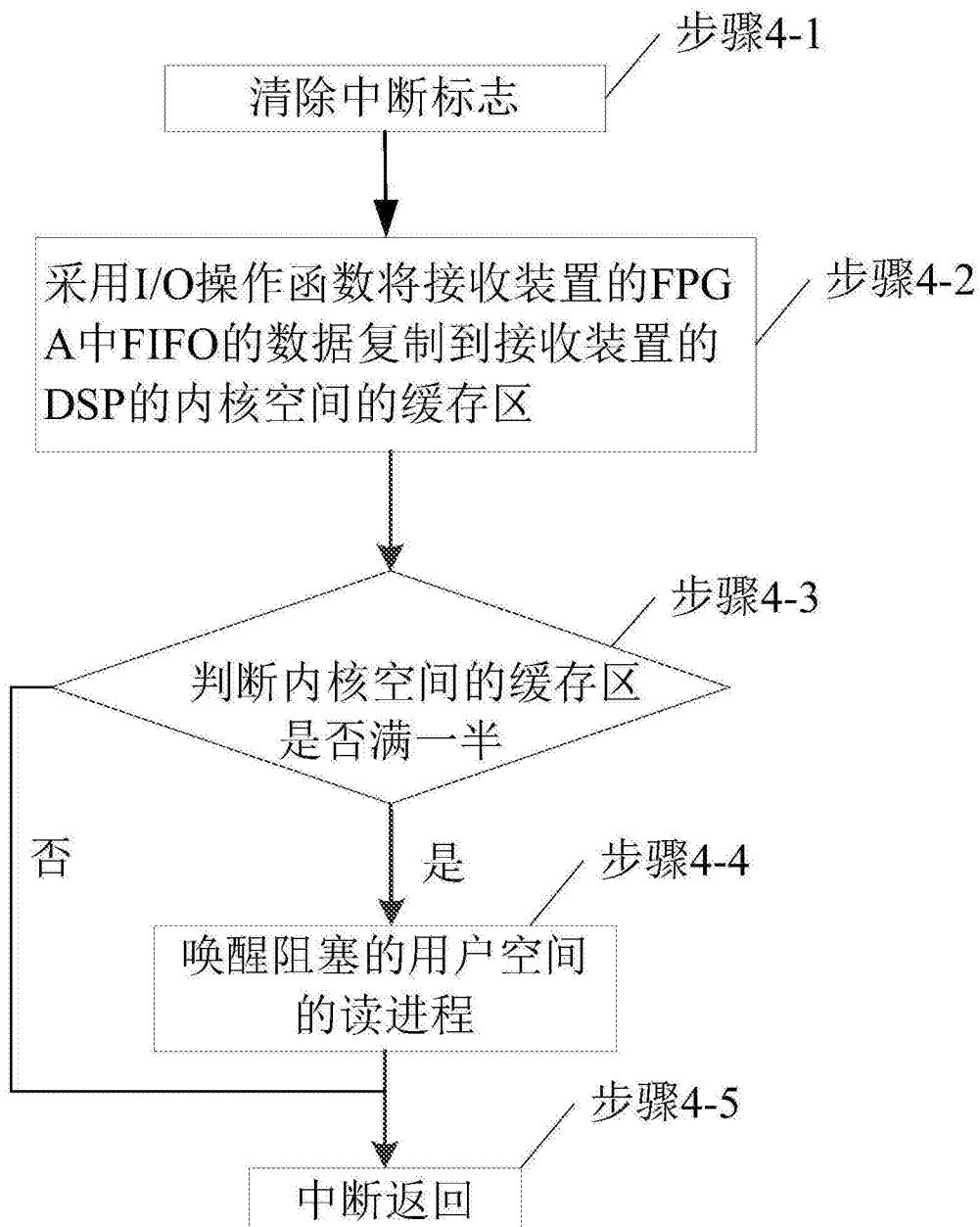


图5

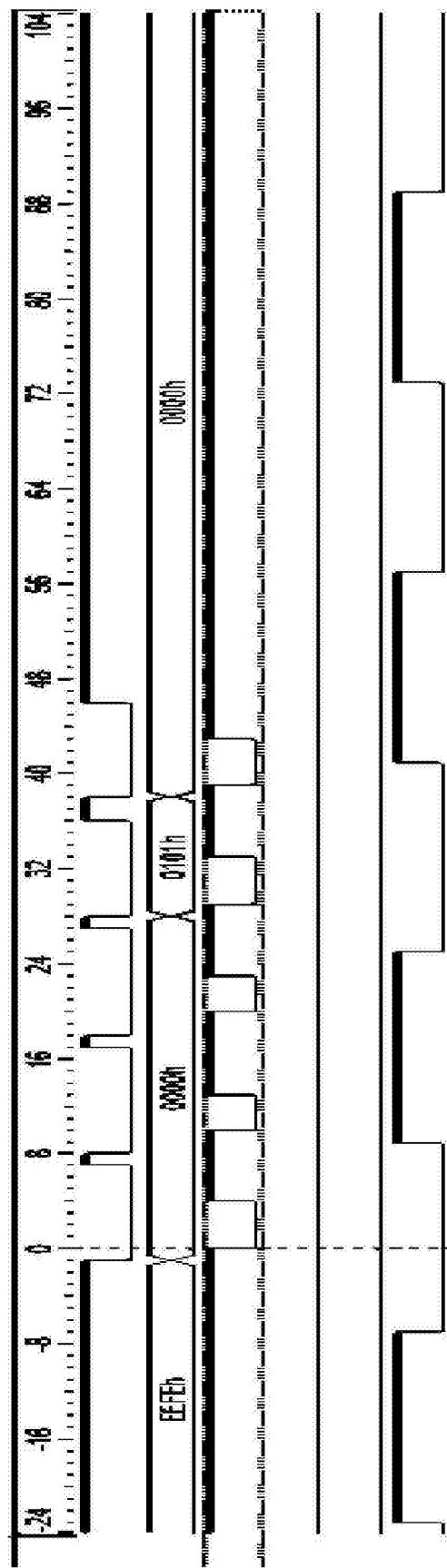


图6

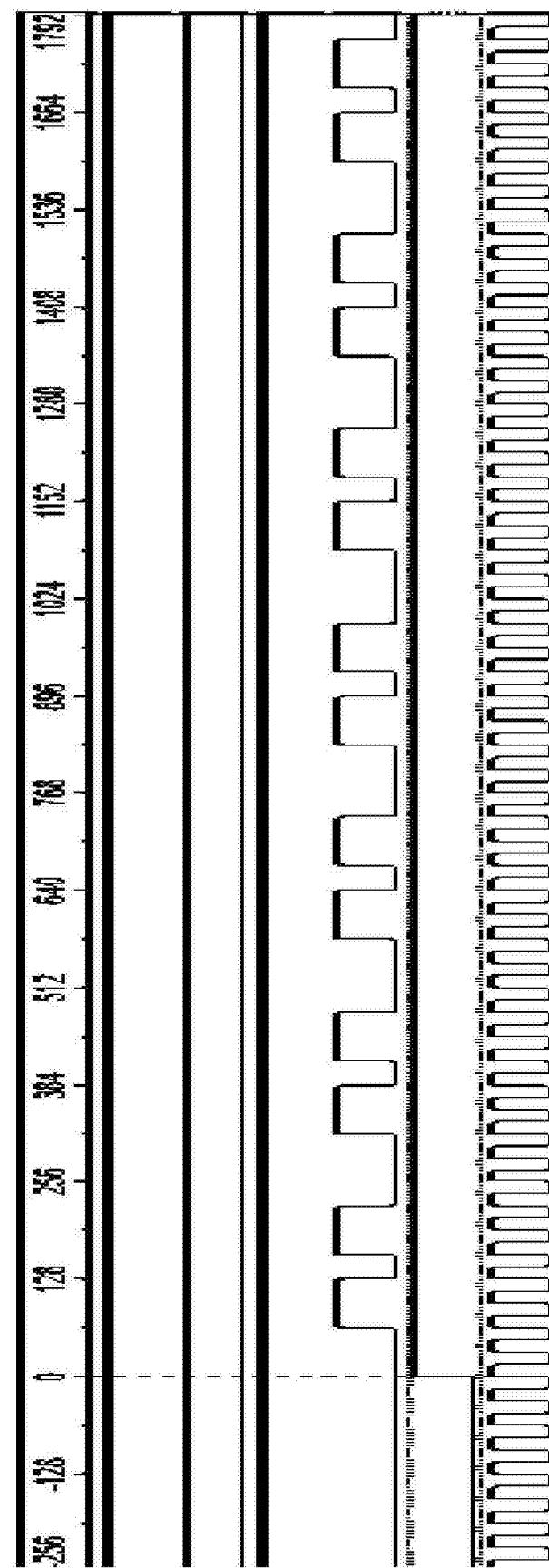


图7

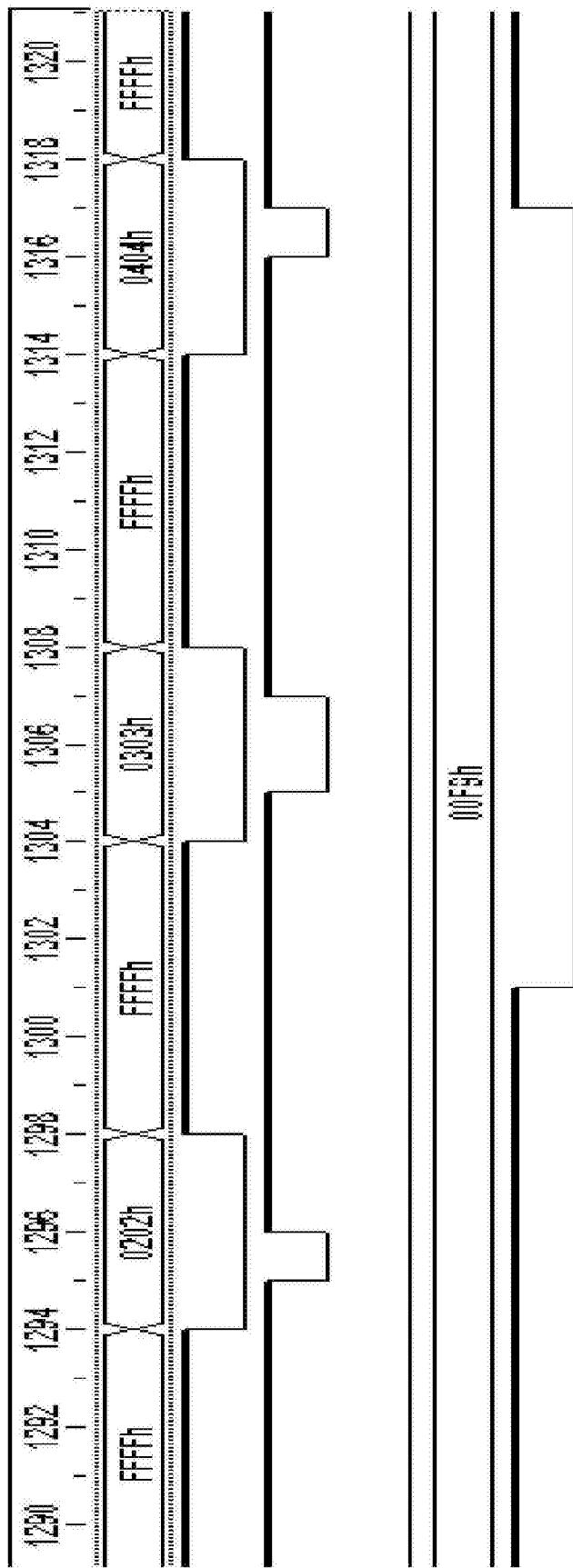


图8